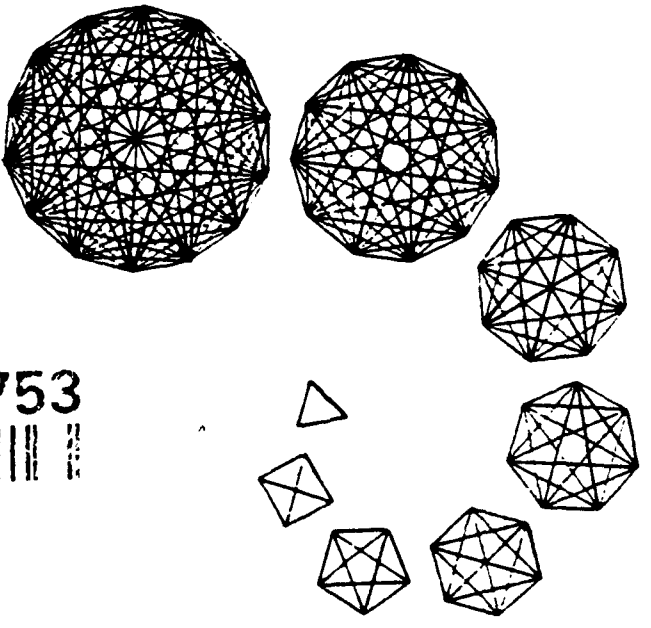
CENTER FOR PURE AND APPLIED MATHEMATICS
UNIVERSITY OF CALIFORNIA, BERKELEY

PAM- 554

Accurate Singular Values and Differential qd Algorithms

K. Vince Fernando, and Beresford N. Parlett

July 1992



This report was done with support from the Center for Pure and Applied Mathematics. Any conclusions or opinions expressed in this report represent solely those of the author(s) and not necessarily those of the Center for Pure and Applied Mathematics or the Department of Mathematics.

Accurate Singular Values and Differential qd Algorithms

K Vince Fernando[1,2,a,b] and Beresford N Parlett[3,b]

[1] NAG Ltd, Jordan Hill, Oxford OX2 8DR, UK

[2] Division of Computer Science, University of California, Berkeley, CA 94708, USA

[3] Department of Mathematics, University of California, Berkeley, CA 94720, USA

July 11, 1992

This is dedicated to the memory of
Heinz Rutishauser

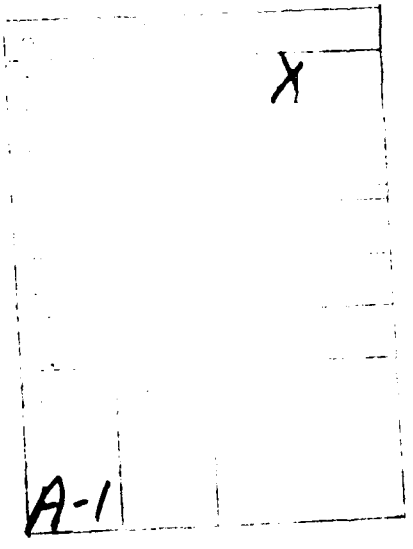
[a]Supported by NSF, under grant ASC-9005933
[b]Supported by ONR, contract N000014-90-J-1372

Abstract

We have discovered a new implementation of the qd algorithm that has a far wider domain of stability than Rutishauser's version. Our algorithm was developed from an examination of the *LR-Cholesky transformation* and can be adapted to parallel computation in stark contrast to traditional qd. Our algorithm also yields useful a posteriori upper and lower bounds on the smallest singular value of a bidiagonal matrix.

The zero-shift bidiagonal QR of Demmel and Kahan computes the smallest singular values to maximal relative accuracy and the others to maximal absolute accuracy with little or no degradation in efficiency when compared with the LINPACK code. Our algorithm obtains maximal relative accuracy for all the singular values and runs at least four times faster than the LINPACK code.

Key words: qd, LR algorithm, Cholesky decomposition, singular values, SVD, bidiagonal matrices



Contents

1 Introduction and Summary

In September 1991 J. W. Demmel and W. M. Kahan were awarded the second SIAM prize in numerical linear algebra for their paper 'Accurate Singular Values of Bidiagonal Matrices' [4], referred to as DK hereafter. Among several valuable results was the observation that the standard bidiagonal QR algorithm used in LINPACK [5], and in many other SVD programs, can be simplified when the shift is zero and, of greater importance, no subtractions occur. The last feature permits very small singular values to be found with (almost) all the accuracy permitted by the data and at no extra cost.

In this paper we show that the DK zero shift algorithm can be further simplified and this simplicity has several benefits. One is that a new algorithm can be implemented in either parallel or pipelined format as an $\mathcal{O}(\log_2 n)$ algorithm. This is pursued in a companion paper [9].

Our investigations began with the modest goal of showing that it was preferable to replace the DK zero-shift QR transform by two steps of zero-shift LR implemented in a qd (quotient-difference) format. Root-free algorithms run considerably faster than standard ones. The surprise here is that to keep the high relative accuracy property it is necessary to use a little known variant of qd (the differential form of the progressive qd algorithm or **dqd** [25], [24]). The standard qd will not suffice as we show in Section 4. There are no subtractions in **dqd**. We suspect that Rutishauser discovered **dqd** in 1968, just two years before his death, and we say more about its history in Sections 4 and 11.

What we want to stress here is that, for reasons we may never know, Rutishauser did not consider the shifted version of **dqd**. Incidentally this differential qd is not to be confused with the continuous analogue of qd (see [21]) and more recent work on QR flows. The trouble with the shifted version of the ordinary qd algorithm is that it cannot recover from a shift that is too large. Consequently qd algorithms have been shackled with very conservative shift strategies, such as Newton's method, and earned the reputation of being slow compared to the QR algorithm. Had Rutishauser considered shifts with differential qd (**dqds** hereafter) he would have realized, as we soon did, that the transformation may be split into two parts. The parts depend on whether the machine is of sequential or parallel type but, in each case, a shift that is too big reveals itself before the old matrix is overwritten and so need not be invoked. An unused shift is not wasted because it gives an improved upper bound on the smallest singular value at a cost less than one qd transformation as well as contributing to an improved shift.

Our approach frees the algorithm to exploit powerful shift strategies while preserving high relative accuracy all the time. In contrast the QR algorithm delivers high relative accuracy only with a zero shift.

Even though our algorithms must find the singular values in order we can use shift strategies that are at least quadratically convergent. This is better than fourth order convergence for QR. When only the smallest few singular values are needed this ordering constraint is a great advantage. Another rather subtle feature is that it is not necessary

to make an extra $\mathcal{O}(n)$ check for splitting of the matrix into a direct sum. The necessary information is provided by the auxiliary quantities.

In June 1992 we discovered that our **dqds** algorithm enjoys high relative stability for all shifts provided that they avoid underflow, overflow or divide by zero. Consequently it can be used in a variety of applications (eigenvalues of symmetric or unsymmetric tridiagonals, zeros of polynomials, poles and zeros of transfer functions and many applications involving continued fractions) where Rutishauser's **qd** has been abandoned because of its instability in the general case.

Our error bounds for singular values are significantly smaller than those in DK and the approach is quite transparent. It was this analysis that showed us the possibility of violating positivity while still maintaining maximal relative accuracy for all singular values, not just the small ones.

It gradually dawned on us as we developed the algorithm that we were breaking away from the *orthogonal paradigm* that has dominated the field of matrix computations (called numerical linear algebra by highbrows) since the 1960's. It seems to be sacrilegious to be achieving greater accuracy and on average, a four fold speed-up[1] by simply abandoning QR for something equivalent to LR. See Section 9.3 for details. High accuracy comes from the fact that **dqds** spends most of its time transforming lower triangular 2×2s into upper triangular 2×2s by premultiplication.

Rutishauser gave no direct explanation for the way shifts are introduced into **qd**. We have supplied one in terms of matrix factorizations in Section 5.1 and go on to list the possible choices for a shift in Section 6 and 9.

Section 3 presents the unifying general result which shows that it is possible to implement the LR-Cholesky algorithm of Rutishauser [22], [26] using orthogonal transformations only. Perhaps this is the key idea exploited in the paper. Since the term LR-Cholesky over describes the algorithm we simply refer to it as the Cholesky Algorithm. Our orthogonal Cholesky algorithm is applicable to dense matrices; this more general case is studied elsewhere [8].

We want to point out the unusual historical lineage of this algorithm. The qd algorithm begat the LR algorithm which then gave rise to the QR algorithm of Francis. This in turn led to the Golub-Kahan and Golub-Reinsch algorithms for singular values of bidiagonal matrices which lead to the DK zero-shift variant. This inspired our orthogonal algorithm of which differential qd is the root-free version. We are back to qd again but with a new implementation.

As a service to the busy readers we have included a brief account of the origins of qd and a summary of the DK paper. When reading [25] we regretted that the link between continued fractions and our matrices was not made explicit. We provide the connection

[1] All our computations are performed on a DECstation 5000/120 using double precision arithmetic (53-bit mantissa).

in the final section.

2 Notation and Normalization

This paper does not involve vectors very much and so we do not follow Householder conventions. However capital roman letters denote matrices while lower case Roman and Greek letters denote scalars. On the rare occasions when a vector is needed it is denoted by a lower case roman letter in boldface.

As usual the singular values of an $n \times n$ matrix C are arranged in monotone decreasing order and denoted by $\sigma_1, \sigma_2, \ldots, \sigma_n$, their union is $\sigma[C]$.

- We make reference to the QR factorization of a matrix. This is the matrix form of the Gram-Schmidt orthonormalizing process applied to the columns of the matrix in natural order. By convention the diagonal entries of the upper triangular factor R are taken nonnegative. See Golub and Van Loan [11] for details.
- We make reference to the Cholesky factorization of a positive definite matrix into the product of a lower triangular matrix and its conjugate transpose. The factors are unique.
- We make references to the LR and QR algorithms. These are defined in the appropriate places.

We shall be concerned mainly with bidiagonal matrices which we call B and take them to be *upper* bidiagonal. To save space we write the bidiagonal matrix

$$B = \begin{bmatrix} a_1 & b_1 & & & & \\ & a_2 & b_2 & & & \\ & & \cdot & \cdot & & \\ & & & \cdot & \cdot & \\ & & & & a_{n-1} & b_{n-1} \\ & & & & & a_n \end{bmatrix}$$

as

$$B = bidiag \left\{ \begin{matrix} & b_1 & & b_2 & & \cdot & & b_{n-2} & & b_{n-1} & \\ a_1 & & a_2 & & \cdot & & \cdot & & a_{n-1} & & a_n \end{matrix} \right\}.$$

2.1 Normalization

Consider now the effect of a zero value among the parameters of $n \times n$ bidiagonal B.

2.1.1 Superdiagonal

Suppose that $b_k = 0$, $k < n$. Then B may be written as a direct (or diagonal) sum of two bidiagonals B_1 and B_2. Moreover

$$\sigma[B] = \sigma[B_1] \cup \sigma[B_2].$$

This case makes the calculation of singular values easier. Even more important is the fact that our algorithms do not suffer from the failure to detect such a split when it occurs. However, the transition from a linearly convergent shift to a quadratic shift will not occur if the split lies undetected for too long.

2.1.2 Diagonal

Let $a_k = 0$, $k < n$. Since $|\det B| = \prod_{i=1}^{n} |a_i| = \prod_{i=1}^{n} \sigma_i$ it follows that $\sigma_n = 0$. However some work is needed in order to *deflate* this value, i.e. to find a new B of order $n-1$ yielding the remaining singular values of B. In exact arithmetic one iteration of any of the unshifted algorithms given later is guaranteed to produce the desired B and so this case does not need special treatment. The zero diagonal entry may be driven to the closest end of the matrix.

If $a_k = 0$, $k < n$, at one step of our algorithm and if $a_n = 0$ at the next step then b_{k-1} will also vanish and so produce a split into two bidiagonals.

2.1.3 Signs

If the matrix is real, then using pre and post multiplication by matrices of the form $diag\{\pm 1\}$ any sign pattern may be imposed on the entries of B without changing the singular values. If the matrix is complex, then it could be transformed to a real matrix by pre and post multiplication by matrices of the form $diag\{\exp(i\omega)\}$ where $i^2 = -1$ and ω is real.

There is little loss of generality in assuming, when necessary, that B is of real positive type; all its parameters exceed 0. However in Section 5.3 we address the practical question of when to relax the requirement of positivity.

3 Orthogonal Form of the Cholesky Algorithm

The result given in the theorem below is implicit in proofs that one step of the QR algorithm is equal to two steps of the Cholesky algorithm. Nevertheless it appears not to have been stated explicitly before and was not known to several experts whom we consulted. So for the next few paragraphs we consider full complex matrices. Recall that the Cholesky factorization of a positive definite Hermitian matrix $A(= A^*)$ may be written as $A = LL^*$ where L is lower triangular.

Definition. The Cholesky *transform* of $A = LL^*$ is

$$\hat{A} := L^*L$$

The Cholesky *algorithm*, consisting of successive applications of the Cholesky transformation, is a special case of the LR algorithm.

We now consider the relation between L and $\hat{L}$, the Cholesky factors of A and $\hat{A}$, respectively.

Theorem 1 *Let $\hat{A} = \hat{L}\hat{L}^*$ be the Cholesky factorization of the Cholesky transform of positive definite $A = LL^*$. Then*

$$L = Q\hat{L}^*$$

is the QR factorization of L.

Some may prefer the formulation

$$R^* = Q\hat{R}$$

with $A = R^*R$ and $\hat{A} = \hat{R}^*\hat{R}$.

Proof. Since A is positive definite all factors mentioned below are unique. By definition of $\hat{L}$

$$L^*L = \hat{L}\hat{L}^*.$$

We seek invertible F such that

$$L = F\hat{L}^*, \tag{1}$$

$$L^* = \hat{L}F^{-1}. \tag{2}$$

Transpose and conjugate (1) and use invertibility of $\hat{L}$ in (2) to find

$$F^* = \hat{L}^{-1}L^* = F^{-1}.$$

So F is unitary and since $\hat{L}^*$ is upper triangular with positive diagonal Equation (1) above gives the QR factorization of L, as claimed. •

The theorem shows that $\hat{L}$ may be obtained from L by orthogonal transformations without forming $\hat{A}$. Moreover just as QR may be performed with column pivoting so can we obtain the Cholesky factor of a permutation of $\hat{A}$. A general application of Theorem 1 is presented in Fernando and Parlett [8] but here we return to the bidiagonal case.

The basic equation $\hat{L}\hat{L}^t = L^tL$ guarantees that the Cholesky algorithm preserves bandwidth. In particular, bidiagonal B gives rise to tridiagonal $A = B^tB$ and a bidiagonal $\hat{B}$. In order to study how $\hat{B}$ is derived from B, let

$$B = bidiag\left\{ \begin{array}{ccccccccc} & b_1 & & b_2 & & \cdot & b_{n-2} & & b_{n-1} & \\ a_1 & & a_2 & & \cdot & \cdot & & a_{n-1} & & a_n \end{array} \right\}.$$

$$\hat{B} = bidiag\left\{ \begin{array}{ccccccccc} & \hat{b}_1 & & \hat{b}_2 & \cdot & \cdot & \hat{b}_{n-2} & & \hat{b}_{n-1} & \\ \hat{a}_1 & & \hat{a}_2 & & \cdot & \cdot & & \hat{a}_{n-1} & & \hat{a}_n \end{array} \right\}.$$

where $\hat{B}^t\hat{B} = BB^t$. By Theorem 1

$$B^t = Q\hat{B}.$$

The matrix Q may be written as a product of $(n-1)$ plane rotation matrices [11].

$$Q = G_1G_2 \ldots G_{n-1}.$$

Before the annihilation of the subdiagonal element b_k, the active part of the matrix is of the form,

$$\begin{array}{ccccc} 0 & \hat{a}_{k-1} & \hat{b}_{k-1} & & \\ & 0 & \tilde{a}_k & 0 & \\ & & b_k & a_{k+1} & 0 \\ & & & b_{k+1} & a_{k+2} \end{array} \qquad (3)$$

and after the plane rotation G_k^t, the matrix becomes

$$\begin{array}{ccccc} 0 & \hat{a}_{k-1} & \hat{b}_{k-1} & & \\ & 0 & \hat{a}_k & \hat{b}_k & \\ & & 0 & \tilde{a}_{k+1} & 0 \\ & & & b_{k+1} & a_{k+2} \end{array} \qquad (4)$$

Formally we may set $B^{(0)} = B^t$ and, for $k = 1, \ldots, n-1$

$$B^{(k)} = G_k^t B^{(k-1)}. \qquad (5)$$

Finally $\hat{B} = B^{(n-1)}$ and, from (3) and (4), with $\tilde{a}_1 = a_1$ and $c_k^2 + s_k^2 = 1$,

$$\begin{aligned} \hat{a}_k &= \sqrt{\tilde{a}_k^2 + b_k^2} = \tilde{a}_k / c_k && (6) \\ s_k &= b_k / \hat{a}_k \\ c_k &= \tilde{a}_k / \hat{a}_k && (7) \\ \hat{b}_k &= s_k a_{k+1} = b_k a_{k+1} / \hat{a}_k && (8) \\ \tilde{a}_{k+1} &= c_k a_{k+1} = \tilde{a}_k a_{k+1} / \hat{a}_k. \end{aligned}$$

There is some redundancy in the equations given above but their most important property is the absence of subtractions. This ensures high relative accuracy in the new entries $\hat{a}_i$ and $\hat{b}_i$. Observe that neither s_k nor c_k is needed explicitly to compute the new entries. To the best of our knowledge the algorithm given below is new. For reasons that appear in the next section we call it the *Orthogonal qd-Algorithm* or **oqd**. It is convenient to use

$$cabs(x, y) = \sqrt{x^2 + y^2} \qquad (9)$$

whose name stands for the complex *abs*olute value of $x + iy$. In numerical computing (e.g. Eispack), an alternative name for *cabs* is *pythag*.

Algorithm 1 (oqd)

$\tilde{a} := a_1$
for $k = 1, n-1$
  $\hat{a}_k := cabs(\tilde{a}, b_k)$
  $\hat{b}_k := b_k * (a_{k+1} / \hat{a}_k)$
  $\tilde{a} := \tilde{a} * (a_{k+1} / \hat{a}_k)$
end for
$\hat{a}_n := \tilde{a}$

This algorithm will undergo several transformations in the following pages before we are ready to implement it. Nevertheless, even at this stage, two applications of it are slightly better (fewer multiplications) than the DK Zero Shift QR algorithm [4] described briefly in our Section 10.

The inner loop comparisons given in Table 1 are based on one QR step which is equal to two LR steps. We have taken into account the common sub-expression $a_{k+1}/\hat{a}_k$ in the estimation of the complexity of **oqd** (Algorithm 1).

DK uses six auxiliary variables while **oqd** needs only one. The memory traffic is essentially determined by the number of variables, arithmetic operations and assignment statements. In most advanced architectures, memory access is more expensive than floating-point operations and in such machines the **oqd** will be very advantageous because of fewer read and write operations.

	DK	oqd
cabs	2	1*2
divisions	2	1*2
multiplications	6	2*2
conditionals	1	0
assignments	7	3*2
auxiliary variables	6	1

Table 1: Complexity of Demmel-Kahan and **oqd**

4 The Quotient Difference Algorithm

It is easy to avoid taking the square roots that appear in **oqd** (Algorithm 1) . Define $b_n := 0$ and $q_k = a_k^2$, $e_k = b_k^2$, $k = 1, 2, \ldots, n$. By simply squaring each assignment in **oqd** (Algorithm 1) one obtains an algorithm that turns out to be a little known variant of the quotient difference algorithm. Rutishauser developed his qd algorithm in several papers from 1953 or 1954 (e.g. [20]) until his early death in 1970 but this variant appeared in English only in 1990 in [25] which is the translation of the German original [24] published in 1976. The full list of the papers on qd by Rutishauser can be found in the above mentioned books which were published posthumously.

In the notes at the end of [20] and at the end of volume 2 of [24] this variant is called the *differential form* of the progressive qd algorithm or **dqd**. These notes were based on unfinished manuscripts of Rutishauser.

Algorithm 2 (dqd)

$$
\begin{aligned}
&d := q_1 \\
&\textit{for } k := 1, n-1 \\
&\quad \hat{q}_k := d + e_k \\
&\quad \hat{e}_k := e_k * (q_{k+1}/\hat{q}_k) \\
&\quad d := d * (q_{k+1}/\hat{q}_k) \\
&\textit{end for} \\
&\hat{q}_n := d
\end{aligned}
$$

The implementation of **dqd** (Algorithm 2) requires only 1 division, 2 multiplies, and 1 addition in the inner loop. No subtractions occur.

The intermediate variable d may be removed. At step k, $d = d_k$ and the trick is to write it as a difference.

$$d_{k+1} = c_k^2 q_{k+1} = q_{k+1} - s_k^2 q_{k+1} = q_{k+1} - \hat{e}_k.$$

Algorithm 3 (qd)

$$\hat{e}_0 = 0$$
$$\textit{for } k := 1, n-1$$
$$\quad \hat{q}_k := (q_k - \hat{e}_{k-1}) + e_k$$
$$\quad \hat{e}_k := e_k * q_{k+1}/\hat{q}_k$$
$$\textit{end for}$$
$$\hat{q}_n := q_n - \hat{e}_{n-1}$$

Table 2 compares the complexity of orthogonal, differential and standard qd algorithms.

	oqd	dqd	qd
cabs	1	0	0
divisions	2	1	1
multiplications	4	2	1
additions	1	1	1
subtractions	0	0	1
assignments	3	3	2
auxiliary variables	1	1	0

Table 2: Complexity of **oqd**, **dqd** and **qd**

We hasten to add that Rutishauser did not derive the qd algorithm from our Theorem 1 but from ideas described in Section 11.

For positive B, **dqd** and **qd** are stable in the sense that all intermediate quantities are bounded by $\|B\|^2$. Singular value errors provoked by finite precision arithmetic will be tiny compared to σ_1^2. This is satisfactory for many purposes and it was not generally appreciated until the DK paper appeared that bidiagonal matrices do determine all their singular values, however small, to the same relative precision enjoyed by the matrix entries. Since such accuracy can be achieved for little extra cost it seems only right to do so. These considerations lead us to abandon **qd** and concentrate on **dqd** and *oqd*.

Example 1 Here is a bidiagonal Toeplitz matrix with $a_i = 1$, $b_i = 256$ ($q_i = 1$, $e_i = 65536$) for all i. The results of our **dqd** algorithm are given in Table 3. Note that $\sqrt{q_{64}} = 1.9093060930437717 \times 10^{-152} \approx 2^{-504}$ gives σ_{64} correct to full machine precision.

The results for **qd** were identical to **dqd** except that the crucial element q_{64} became zero in both steps. Hence **qd** is not suitable for computation of small singular values with high relative accuracy. •

Example 2 We have rerun Example 1 but with a smaller value of ($n = 5$) and the results are given in Table 4. For this example, $\sigma_5 = \sqrt{q_5} = 2.3282709094019085 \times 10^{-10}$ which is correct to full machine precision. For comparison, the answer given by the LINPACK SVD routine **dsvdc** (which is based on the Golub-Reinsch algorithm) is $2.3282704794711363 \times 10^{-10}$ which gets 7 of the 15 digits correct.

	after the first pass	after the second pass
q_1	6.5537000000000000D+04	6.5537999984741444D+04
q_2	6.5536000015258556D+04	6.5536000061032595D+04
q_3	6.5536000000000233D+04	6.5536000000001397D+04
q_4 to q_{63}	6.5536000000000000D+04	6.5536000000000000D+04
q_{64}	3.6455053829317361D-304	3.6454497569340717D-304
e_1	9.9998474144376459D-01	9.9995422572819948D-01
e_2	9.9999999976717291D-01	9.9999999883589297D-01
e_3	9.9999999999999645D-01	9.9999999999997513D-01
e_4 to e_{62}	1.0000000000000000D+00	1.0000000000000000D+00
e_{63}	1.0000000000000000D+00	5.5625997664363648D-309

Table 3: Numerical results for Example 1

Using **qd** we got almost identical results except that q_5 is zero in both sweeps. Thus, σ_5 is zero according to the **qd** algorithm. Thus, **qd** does not deliver as much accuracy as Golub-Reinsch; in fact it can be shown that **qd** sometimes delivers zero for singular values as large as $\sqrt{macheps} * \|B\|$. •

	after the first pass	after the second pass
q_1	6.5537000000000000D+04	6.5537999984741449D+04
q_2	6.5536000015258551D+04	6.5536000061032593D+04
q_3	6.5536000000000238D+04	6.5536000000001395D+04
q_4	6.5536000000000000D+04	6.5536000000000000D+04
q_5	5.4209281443662679D-20	5.4208454275671899D-20
e_1	9.9998474144376457D-01	9.9995422572819948D-01
e_2	9.9999999976717293D-01	9.9999999883589292D-01
e_3	9.9999999999999642D-01	9.9999999999997509D-01
e_4	1.0000000000000000D+00	8.2716799077854419D-25

Table 4: Numerical results for Example 2

Some people do not like root free algorithms (e.g. **dqd**) because they limit the domain of the matrices to which they can be applied. For example, a bidiagonal B whose singular values vary from 10^{30} to 10^{-30} could be diagonalized in single precision on an IBM machine by **oqd** (Algorithm 1) but not by **dqd** (Algorithm 2) because of overflow and underflow.

We conclude this section by pointing out that **qd** (Algorithm 3), the standard qd algorithm, consists of the so-called *rhombus rules* arranged in computational form and these rules are a direct consequence of the defining equation

$$BB^t = \hat{B}^t \hat{B}.$$

Equate the (k,k) entry on each side to obtain

$$a_k^2 + b_k^2 = \hat{b}_{k-1}^2 + \hat{a}_k^2 \tag{10}$$

$$q_k + e_k = \hat{e}_{k-1} + \hat{q}_k.$$

and equate the $(k, k+1)$ entry on each side to obtain

$$b_k a_{k+1} = \hat{a}_k \hat{b}_k \tag{11}$$

$$e_k q_{k+1} = \hat{q}_k \hat{e}_k.$$

The rhombus rules can be also derived from $B^t = Q\hat{B}$ by noting that orthogonal transformation Q changes neither the norms nor the inner products of the columns. The reason for the name rhombus rule is indicated in Figure 3 of Section 11.

5 Incorporation of Shifts

Rutishauser introduced shifts into the **qd** almost from the beginning and we could simply quote him. Unfortunately he does not give any explanation of how he derived the appropriate modification of **qd** (given in Section 4). So we provide one at the end of Section 5.1.

5.1 Shifted qd Algorithms

In eigenvalue calculations, shifts are natural and can be easily incorporated since

$$\lambda(A - \tau^2 I) = \lambda(A) - \tau^2$$

where τ^2 is the shift and $\lambda(A)$ indicates an eigenvalue of A. Thus, by subtracting τ^2 from the diagonals of the matrix, we can introduce origin shifts into the Cholesky algorithm.

A shift τ can be introduced into **oqd** (Algorithm 1, Section 3) by modifying statements involving $\hat{a}$ and $\tilde{a}$.

Algorithm 4 (oqds)

$\tilde{a} := a_1$
for $k = 1, n-1$
  $\hat{a}_k := \sqrt{\tilde{a}^2 + b_k^2 - \tau^2}$
  $\hat{b}_k := b_k * (a_{k+1}/\hat{a}_k)$
  $\tilde{a} := \sqrt{\tilde{a}^2 - \tau^2} * (a_{k+1}/\hat{a}_k)$
end for
$\hat{a}_n := \sqrt{\tilde{a}^2 - \tau^2}$

It may be verified that $\hat{B}^t\hat{B} = BB^t - \tau^2 I$. To keep $\hat{B}$ real the shift must satisfy

$$\tau \leq \sigma_n[B] \tag{12}$$

but this constraint is not formally necessary for **dqd** (Algorithm 2) which uses

$$\hat{q}_k := d_k + e_k - \tau^2.$$

Algorithm 5 (dqds)

$$\begin{aligned}
&d := q_1 - \tau^2 \\
&\textit{for } k := 1, n-1 \\
&\quad \hat{q}_k := d + e_k \\
&\quad \hat{e}_k := e_k * (q_{k+1}/\hat{q}_k) \\
&\quad d := d * (q_{k+1}/\hat{q}_k) - \tau^2 \\
&\textit{end for} \\
&\hat{q}_n := d
\end{aligned}$$

The constraint (12) is also unnecessary for **qd**.

Algorithm 6 (qds)

$$\begin{aligned}
&\hat{e}_0 = 0 \\
&\textit{for } k := 1, n-1 \\
&\quad \hat{q}_k := (q_k - \hat{e}_{k-1}) + e_k - \tau^2 \\
&\quad \hat{e}_k := e_k * q_{k+1}/\hat{q}_k \\
&\textit{end for} \\
&\hat{q}_n := q_n - \hat{e}_{n-1} - \tau^2
\end{aligned}$$

All that is lacking is an analogue of the orthogonal connection (Theorem 1)

$$B^t = Q\hat{B}.$$

For that it is necessary to abandon square matrices and write

$$\begin{bmatrix} B^t \\ O \end{bmatrix} = Q \begin{bmatrix} \hat{B} \\ \tau I \end{bmatrix}.$$

The new Q is $2n \times 2n$ and is not unique. However its first n rows are uniquely determined by B and τ for $\tau \leq \sigma_n[B]$.

It is at this point that the superiority of the qd formulation becomes clear. DK showed that the standard Golub-Reinsch bidiagonal QR algorithm may be simplified when the shift is zero; see Section 10 for the details. Our algorithms (1,2, or 3) are already simpler than the DK zero shift QR and they also permit use of a non-zero shift with no impediment to pipelined or parallel implementation or high relative accuracy. See [9] for details. This is strong evidence that our formulation is the natural one.

5.2 The Two Phase Implementation

At first sight the auxiliary quantities d_i, $i = 1, \ldots, n$ that occur in **dqd** are seen as the price to be paid for securing high relative accuracy. On further consideration they may be seen as an attractive feature that permits an aggressive shift strategy that also preserves high relative accuracy in the computed singular values. Moreover, as an extra bonus, we find that the vector $d = (d_1, \ldots, d_n)$ may be computed in $\mathcal{O}(\log_2 n)$ steps in a parallel computer using the technique called parallel prefix operation in computer science writings, see [3].

Consider next the implementation of **dqds**. The auxiliary quantities d_i may be computed prior to any modification of q and e since

$$\begin{aligned} d_{k+1} &= d_k q_{k+1}/\hat{q}_k - \tau^2 \\ &= d_k q_{k+1}/(d_k + e_k) - \tau^2. \end{aligned} \tag{13}$$

An alternative formulation is

$$d_{k+1} = \frac{q_{k+1}}{1 + e_k/d_k} - \tau^2 \tag{14}$$

but a division costs more than a multiplication.

It is at this point that one sees the advantage of arithmetic units that conform to the ANSI/IEEE floating point standard 754: there is no need to test at each instance of (13) or (14) to prevent division by zero. The occurence of a k with $d_k = \infty$ does no harm. It signals that

$$\sigma_n^2[B] < \tau^2$$

and the transformation of B to $\hat{B}$ (Phase 2) should not be completed. The effort in running (13) is not wasted because it yields a new upper bound on $\sigma_n^2[B]$.

Using (13), $d_k = \infty$ yields $d_{k+1} = \infty/\infty = NaN$ (not a number) and then $\hat{q}_i = NaN$ for $i > k+1$. Using (14), $d_k = \infty$ yields $d_{k+1} = q_{k+1} - \tau^2$ which is a better answer.

5.3 Almost Positive Bidiagonals

The standard qd algorithm is well defined for most shifts but it may not be stable in an absolute sense; i.e. the new array $\{\hat{q}, \hat{e}\}$ may be far greater than old one $\{q, e\}$. Rutishauser proved stability under the assumption of positivity and took great care in his implementation to preserve this property.

Our **dqds** algorithm has the advantage of maintaining relative stability in the positive case and, fortunately, even beyond. We currently impose the requirement

$$\tau^2 < \frac{1}{2}\sigma_{n-1}^2[B_{n-1}] + e_{n-1}$$

where B_{n-1} is the leading principal submatrix of B_n because it ensures that the only entries in $\{\hat{q}, \hat{e}\}$ that could go negative are $\hat{e}_{n-1}$ and $\hat{q}_n$. Our goal is to choose τ (actually τ^2) to make $\hat{q}_n$ as small as possible and hence

$$\tau^2 \approx d_n = q_n(1 - e_{n-1}/\hat{q}_{n-1}).$$

Notice how strongly d_n depends on $sign(e_{n-1})$ and $sign(q_n)$ since $\hat{q}_{n-1}$, though unknown, remains positive. There are four possible configurations in the asymptotic regime ($\tau^2 < \frac{1}{2}d_{n-1}+e_{n-1}$) and we designate them by sign pairs: $(sign(e_{n-1}), sign(q_n))$. Each time that **dqds** is invoked there is no doubt about $sign(\hat{e}_{n-1})$ but $sign(\hat{q}_n)$ will not be predictable since the aim is to have $\hat{q}_n = 0$.

A careful study of the last three assignments in **dqds** shows the following possible paths the iteration could follow. Since we do not expect more than 2 steps before convergence (and deflation) some edges may not be traversed.

If $\tau 2 < \sigma_n$

$$\begin{aligned} (+,+) &\longrightarrow (+,+) \\ (+,-) &\longrightarrow (-,+) \\ (-,+) &\longrightarrow (-,-) \\ (-,-) &\longrightarrow (+,+) \end{aligned}$$

If $\tau 2 > \sigma_n$

$$\begin{aligned} (+,+) &\longrightarrow (+,-) \\ (+,-) &\longrightarrow (-,-) \\ (-,+) &\longrightarrow (-,+) \\ (-,-) &\longrightarrow (+,-) \end{aligned}$$

6 Bounds for σ_{min}

6.1 A Posteriori Bounds for the Smallest Singular Value

Our **oqd**(Algorithm 1 in Section 3) transforms B to $\hat{B}$ by making use of n auxiliary quantities $\tilde{a}_k, k = 1, n$. It is possible to give a nice interpretation of the $\tilde{a}_k$ that leads to useful bounds on σ_{min}. This result was also obtained by Rutishauser but his treatment was not based on orthogonal rotations although he knew the matrix interpretation of qd.

If we think of the matrix B^t being transformed into $\hat{B}$ one column at a time in $(n-1)$ little steps then at the end of Step $(k-1)$ row k is a singleton. That is the key technical observation. To describe the situation we refer back to Section 3 and let $Q_k =$

$(G_1 G_2 \ldots G_{k-1})^t$ be the product of the first $(k-1)$ plane rotations used in the reduction process. Thus

$$B^{(k)} = Q_k B^t = \begin{bmatrix} \hat{a}_1 & \hat{b}_1 & & & & & & & \\ 0 & \hat{a}_2 & \hat{b}_2 & & & & & & \\ & 0 & . & . & & & & & \\ & & 0 & \hat{a}_{k-1} & \hat{b}_{k-1} & & & & \\ & & & 0 & \tilde{a}_k & 0 & & & \\ & & & & b_k & a_{k+1} & 0 & & \\ & & & & & b_{k+1} & a_{k+2} & & \\ & & & & & & . & . & 0 \\ & & & & & & & b_{n-2} & a_{n-1} & 0 \\ & & & & & & & & b_{n-1} & a_n \end{bmatrix} \tag{15}$$

Note that $Q_k B^t$ coincides with $\hat{B}$ in rows $1, 2, \ldots, k-1$ and with B^t in rows $k+1, \ldots, n$ while orthogonal Q_k coincides with I_n in rows $k+1, \ldots, n$.

Theorem 2 (Bounds for σ_{min} without shifts) *Apply the* ***dqd*** *transformation to a positive bidiagonal B (see Algorithm 1) to produce $\hat{B}$ and $\tilde{a}_1, \tilde{a}_2, \ldots, \tilde{a}_n$. Then*

1. $\sigma_n \leq \min_k \{\tilde{a}_k\}$
2. $[(BB^t)^{-1}]_{k,k} = \tilde{a}_k^{-2}$
3. $(\sum_{k=1}^n \tilde{a}_k^{-1})^{-1} \leq (\sum_{k=1}^n \tilde{a}_k^{-2})^{-1/2} \leq \sigma_n$.

Proof: Since singular values are invariant under orthogonal transformations and transposition

$$\sigma_n[B] = \sigma_n[Q_k B^t] \leq \|u_k^t Q_k B^t\| = \tilde{a}_k$$

where u_k is the kth column of the identity matrix. The kth row of $Q_k B^t$ is a singleton;

$$u_k^t Q_k B^t = \tilde{a}_k u_k^t.$$

Transposing and rearranging gives

$$\tilde{a}_k^{-1} Q_k u_k = B^{-1} u_k$$

$$\tilde{a}_k^{-2} = (B^{-t} B^{-1})_{k,k}$$

as claimed. Note that

$$\sigma_n^{-2} \leq \sum_{i=1}^n \sigma_i^{-2} = \|B^{-1}\|_F^2 = trace[(BB^t)^{-1}] = \sum_{i=1}^n \tilde{a}_i^{-2}.$$

Finally we get the required result by considering the one and two norms of the vector $(\tilde{a}_1^{-1}, \tilde{a}_2^{-1}, \ldots \tilde{a}_n^{-1})$. •

We can compute bounds on $\sigma_{min}[B]$ even when the algorithm is used with shifts τ provided that $\tau \leq \sigma_{min}[B]$. Formally the reduction

$$\begin{pmatrix} B^t \\ 0 \end{pmatrix} \rightarrow \begin{pmatrix} \hat{B} \\ \tau I \end{pmatrix}$$

requires $2(n-1)$ plane rotations (not just $n-1$) because the rotation G_i in $(i, i+1)$ must be preceded by a rotation $\bar{G}_i$ in plane $(i, n+i)$ in order to introduce τ into position $(n+i, i)$. Thus the rows $1, \ldots, k-1$ and $n+1, \ldots, n+k-1$ of

$$Q_k \begin{pmatrix} B^t \\ 0 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} \hat{B} \\ \tau I \end{pmatrix}$$

are coincident. Also the rows $k+1, \ldots, n$ and $n+k, \ldots, 2n$ of

$$Q^t \begin{pmatrix} B^t \\ 0 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} B^t \\ 0 \end{pmatrix}$$

are the same. However row k is still a singleton in fact

$$u_k^t Q_k \begin{pmatrix} B^t \\ 0 \end{pmatrix} = \tilde{a}_k u_k^t. \tag{16}$$

Theorem 3 (Bounds for σ_{min} with shifts) *If the* ***dqds*** *algorithm with shift τ transforms positive bidiagonal B into positive $\hat{B}$ with auxiliary quantities $\tilde{a}_1, \ldots, \tilde{a}_n$ then*

1. $\sigma_n \leq \min_k\{\tilde{a}_k\}$
2. $[(BB^t)^{-1}]_{k,k} = \tilde{a}_k^{-2}\|x_k\|^2 < \tilde{a}_k^{-2}$.

where, in (16), $u_k^t Q_k := (x_k^t, y_k^t)$, and x and y each have n entries.

Proof: Since singular values are invariant under orthogonal transformation and transposition,

$$\sigma_n[B] = \sigma_n\left[Q_k \begin{pmatrix} B^t \\ 0 \end{pmatrix}\right] \leq \|u_k^t Q_k \begin{pmatrix} B^t \\ 0 \end{pmatrix}\| = \tilde{a}_k.$$

The last equality uses (16). To establish the second result transpose (16) to obtain

$$\begin{bmatrix} B & 0 \end{bmatrix} Q_k^t u_k = Bx_k = u_k \tilde{a}_k.$$

Since B is invertible,

$$\tilde{a}_k^{-1} x_k = B^{-1} u_k,$$

$$\tilde{a}_k^{-2}\|x_k\|^2 = u_k^t B^{-t} B^{-1} u_k = [(BB^t)^{-1}]_{k,k} \quad \bullet$$

Remark: Since the $\tilde{a}_k$ are monotone decreasing in τ a successful **dqds** transformation produces a better upper bound and a worse lower bound than does **dqd**. Fortunately it is the upper bound that plays an active role in our implementation.

6.2 The Newton shift

The shift strategy used by Bauer to accelerate the rational QR algorithm RATQR is also closely related to part 3 of the above theorem. See [1], [19].

We recall that the Newton shift from 0 for the characteristic polynomial of any matrix A is related to the trace of the inverse. Let

$$\chi_A(t) = det[tI - A] = \prod_{i=1}^{n}(t - \lambda_i).$$

Then, by logarithmic differentiation

$$\frac{\chi'_A(t)}{\chi_A(t)} = \sum_{i=1}^{n} \frac{1}{t - \lambda_i}.$$

In particular

$$\frac{\chi'_A(0)}{\chi_A(0)} = -\sum_{i=1}^{n} \lambda_i^{-1} = -trace A^{-1}$$

because the spectrum of A^{-1} is $\{\lambda_i^{-1}\}_1^n$.

In our case $(\sum \hat{a}_k^{-2})^{-1}$ is the Newton correction from 0 towards σ_n^2.

6.3 The $(1, \infty)$ Bound

The DK paper also provides lower bounds on σ_n. Two recurrences (see Section 10 for details) produce

$$\min_j \lambda_j = \|B^{-1}\|_\infty^{-1}$$

and

$$\min_j \mu_j = \|B^{-1}\|_1^{-1}.$$

Then

$$\sigma_n^{-1} = \|B^{-1}\| \leq \min\{\|B^{-1}\|_\infty^{-1}, \|B^{-1}\|_1^{-1}\}.$$

Since $\|C\| \leq \sqrt{\|C\|_1 \|C\|_\infty}$ for any square matrix C, we can improve the DK bound to give,

$$\sigma_n^{-1} = \|B^{-1}\| \leq \sqrt{\|B^{-1}\|_1 \|B^{-1}\|_\infty} \leq \min\{\|B^{-1}\|_\infty^{-1}, \|B^{-1}\|_1^{-1}\}.$$

6.4 The Johnson Bound

For a general complex matrix C, a Gersgorin-type bound for σ_{min} is given by Johnson (see [14]),

$$\sigma_{min} \geq \max\{0, \theta\}$$

where

$$\theta = \min_i \left\{ |c_{i,i}| - \frac{1}{2} \sum_{k \neq i} |c_{k,i}| + |c_{i,k}| \right\}.$$

For a positive bidiagonal B, this simplifies to

$$\theta = \min_i \left\{ a_i - \frac{1}{2}(b_i + b_{i-1}) \right\}$$

and ultimately this becomes

$$\theta = a_n - \frac{1}{2} b_{n-1}.$$

7 Effects of Finite Precision

7.1 Error Analysis - Overview

One of the benefits of the simplicity of our algorithms **oqd** and **dqd** is that their analysis is relatively easy. The DK zero shift QR transformation, though simpler than the Golub/Reinsch transformation, is complicated enough to defy anything but a forward error analysis. After heroic struggles with innumerable details DK establish the error bound quoted in Section 10.4.

When discussing this result and our own analyses it is convenient to use the acronym *ulp* which stands for *u*nits in the *l*ast *p*lace held. It is the natural way to refer to *relative* differences between numbers. When a result is correctly rounded the error is not more than half an *ulp*. In this section we usually omit the ubiquitous phrase 'at most' qualifying errors and modifications.

Our algorithms still do not admit a pure backward error analysis, the computed output $\hat{B}$ is not the exact output from a matrix very close to B. Nevertheless we can use a hybrid interpretation involving both backward and forward interpretation.

Whereas DK's zero shift guarantees that each computed singular value is in error by no more than $69n^2$ *ulps* our **dqds** algorithm causes no more than $4n$ *ulps* change using any properly chosen shift. However the main point is that our analysis is easy to grasp.

The next subsection establishes this strong property of **dqds**. A similar result holds for **oqds** but the square roots and squaring provoke a slightly larger bound.

The trick of the proof is to define $\vec{B}$ so that the computed auxiliary quantities $\{d_i\}$ are exact outputs of **dqds**. The difference between $\check{B}$ and $\hat{B}$ is the forward error.

At the beginning of the paper we made much of the fact that algorithms **oqd** and **dqd** required no subtractions. Yet, in the interest of efficiency, we have introduced shifts and quietly brought back subtraction. The miracle is that the subtraction is in the d's and does not impair the high relative accuracy property. However **qd** does not guarantee high relative accuracy so long as q's are dominated by neighbouring e's.

Since no intermediate quantities exceed σ_1, it is assumed that the initial data are scaled so that σ_1 (or σ_1^2 for **dqds**) is close to the overflow threshold. Underflow, though possible, is then a rare event.

Finally we remind the reader that the symbol $=$ carries its normal mathematical meaning.

7.2 High Relative Accuracy in the Presence of Shifts

We refer the reader to Section 5.3 where almost positive bidiagonals are introduced. Rutishauser merges the q's and e's into a single array Z:

$$Z := \{q_1, e_1, q_2, e_2, \ldots, e_{n-1}, q_n\}$$

and this is a convenient notation for the analysis which follows.

Before stating our claim we need more notation because the difficulty in the analysis is one of interpretation. Given Z the **dqds** algorithm in finite precision arithmetic produces representable output $\hat{Z}$. We introduce two ideal arrays $\vec{Z}$ and $\check{Z}$ such that $\check{Z}$ is the output of **dqds** with shift τ acting on $\vec{Z}$ *in exact arithmetic.* Moreover $\vec{Z}$ is a small relative perturbation of Z and $\hat{Z}$ is a small relative perturbation of $\check{Z}$. See Figure 1.

Our model of arithmetic is that the floating point result of a basic arithmetic operation $\circ$ satisfies

$$fl(x \circ y) = (x \circ y)(1+\eta) = (x \circ y)/(1+\delta) \tag{17}$$

where η and δ depend on x, y, and $\circ$, and the arithmetic unit but satisfy

$$|\eta| < \epsilon \ , \ |\delta| < \epsilon$$

for a given ϵ that depends only on the arithmetic unit. We shall choose freely the form (η or δ) that suits the analysis.

A fairly simple result is possible because the only truly sequential part of **dqds** is the sequence $\{d_i\}_1^n$. Note that, in exact arithmetic

$$d_{k+1} = \frac{d_k q_{k+1}}{d_k + e_k} - \tau^2$$

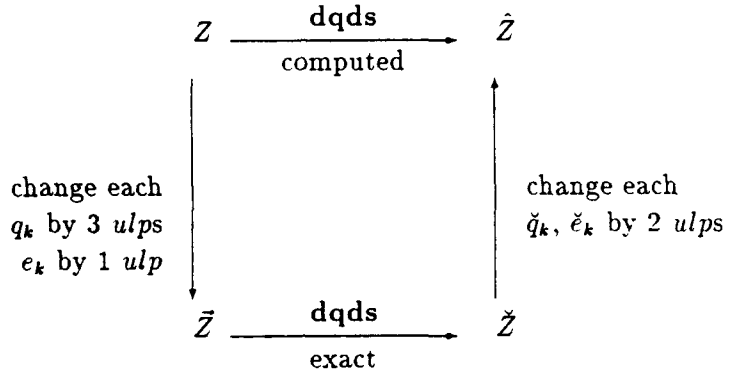


Figure 1: Effects of roundoff

The trick is to write down the relations governing the computed quantities and then to *discern* among them an exact **dqds** transform whose input is close to Z and whose output is close to $\hat{Z}$.

Theorem 4 *In the absence of underflow or overflow, the Z diagram given above commutes and $\check{q}_k$ ($\check{e}_k$) differs from q_k (e_k) by 3 (1) ulps. $\hat{q}_k$ ($\hat{e}_k$) differrs from $\check{q}_k$ ($\check{e}_k$) by 2 (2) ulps.*

Proof: We write down the exact relations satisfied by the computed quantities $\hat{Z}$.

$$\hat{q}_k = (d_k + e_k)/(1+\epsilon_+) \tag{18}$$

$$t_k = q_{k+1}(1+\epsilon_/)/\hat{q}_k = \frac{q_{k+1}(1+\epsilon_/)(1+\epsilon_+)}{d_k + e_k} \tag{19}$$

$$\hat{e}_k = e_k t_k (1+\epsilon_*) = \frac{e_k q_{k+1}(1+\epsilon_/)(1+\epsilon_+)(1+\epsilon_*)}{d_k + e_k} \tag{20}$$

$$d_{k+1} = \frac{\{d_k t_k (1+\epsilon_\star) - \tau^2\}}{1+\epsilon_{k+1}}$$

Note the difference between $*$ and $\star$. Of course all the ϵ's obey (17) and depend on k but we have chosen to single out the one that accounts for the subtraction because it is the only one where the k dependence must be made explicit. In more detail the last relation is

$$(1+\epsilon_{k+1})d_{k+1} = \frac{d_k q_{k+1}}{d_k + e_k}(1+\epsilon_/)(1+\epsilon_+)(1+\epsilon_*) - \tau^2$$

$$= \frac{(1+\epsilon_k)d_k q_{k+1}(1+\epsilon_/)(1+\epsilon_+)(1+\epsilon_*)}{(1+\epsilon_k)d_k + (1+\epsilon_k)e_k} - \tau^2 \tag{21}$$

This tells us how to define $\vec{Z}$. Note that ϵ_k arose in the previous step. Moreover

$$(1+\epsilon_1)d_1 = q_1 - \tau^2 \tag{22}$$

Our choice of $\vec{Z}$, in general, is not a machine representable array.

For $k \geq 1$,

$$\begin{aligned} \vec{d}_k &:= (1+\epsilon_k)d_k \\ \vec{e}_k &:= (1+\epsilon_k)e_k \qquad (23) \\ \vec{q}_{k+1} &:= q_{k+1}(1+\epsilon_/)(1+\epsilon_+)(1+\epsilon_*) \ , \ (\vec{q}_1 = q_1) \qquad (24) \end{aligned}$$

and by (21),

$$\vec{d}_{k+1} = \frac{\vec{d}_k \vec{q}_{k+1}}{\vec{d}_k + \vec{e}_k} - \tau^2 \tag{25}$$

Then, for exact **dqds**, we must define

$$\breve{q}_k := \vec{d}_k + \vec{e}_k = (1+\epsilon_k)(d_k + e_k)$$

$$\breve{e}_k := \frac{\vec{e}_k \vec{q}_{k+1}}{\vec{d}_k + \vec{e}_k}.$$

Finally $\hat{q}_k$ and $\hat{e}_k$ must be recast in terms of $\breve{Z}$;

$$\hat{q}_k = \breve{q}_k/(1+\epsilon_k)(1+\epsilon_+) \quad \text{from (18)} \tag{26}$$

$$\begin{aligned} \hat{e}_k &= \frac{e_k q_{k+1}}{d_k + e_k}(1+\epsilon_/)(1+\epsilon_+)(1+\epsilon_*) \ , \ \text{from (20)} \\ &= \frac{\vec{e}_k \vec{q}_{k+1}}{(\vec{d}_k + \vec{e}_k)} \frac{(1+\epsilon_*)}{(1+\epsilon_*)}. \qquad (27) \end{aligned}$$

It is (23) that yields $\vec{e}_k/(\vec{d}_k + \vec{e}_k) = e_k/(d_k + e_k)$. Equations (23) and (24) give the change from $\breve{Z}$ to $\hat{Z}$, and equation (25) fixes the exact **dqds** transform. •

Recall that, in exact arithmetic, algorithm **dqds** diminishes all eigenvalues (of LR) by the shift. For finite precision execution we have the following.

Corollary 1 *Algorithm* ***dqds*** *preserves high relative stability. When Z and $\hat{Z}$ are positive then the associated bidiagonal matrices B and B ($a_i = \sqrt{q_i}$, $b_i = \sqrt{e_i}$, etc.), together with the associated ideal bidiagonals $\vec{B}$ and $\breve{B}$, satisfy*

$$\sigma_i[\vec{B}] = \sigma_i[B] \exp\{2(n-1)\epsilon_1\},$$

$$\sigma_i^2[\breve{B}] = \sigma_i^2[\vec{B}] - \tau^2,$$

$$\sigma_i[\hat{B}] = \sigma_i[\breve{B}] \exp\{(2n-1)\epsilon_2\},$$

for $i = 1, 2, \ldots, n$, and $\epsilon_1 \leq \epsilon$, $\epsilon_2 \leq \epsilon$.

Proof: For $i = 1, \ldots, n-1$

$$\vec{a}_{i+1} = \sqrt{\vec{q}_{i+1}} = a_{i+1}\sqrt{(1+\epsilon_/)(1+\epsilon_+)(1+\epsilon_*)}$$

$$\vec{b}_i = \sqrt{\vec{e}_i} = b_i\sqrt{(1+\epsilon_i)}.$$

By Theorem 2 in DK, the relative change in any singular value in going from B to $\vec{B}$ is the product of all the relative changes, namely

$$\prod_{i=1}^{n-1}[(1+\epsilon_/)(1+\epsilon_+)(1+\epsilon_*)(1+\epsilon_i)]^{\frac{1}{2}} \leq \exp 2(n-1)\epsilon.$$

Similarly

$$\hat{a}_i = \sqrt{\hat{q}_i} = \check{a}_i/\sqrt{(1+\epsilon_i)(1+\epsilon_+)} \ , \ i < n$$

$$\hat{b}_i = \sqrt{\hat{e}_i} = \check{b}_i\sqrt{(1+\epsilon_*)(1+\epsilon_*)} \ , \ i < n$$

$$\hat{a}_n = \sqrt{\hat{d}_n} = \check{a}_n/\sqrt{(1+\epsilon_n)}.$$

The relative change in any singular value in the transformation from $\check{B}$ to $\hat{B}$ is bounded by

$$\sqrt{1+\epsilon_n}\prod_{i=1}^{n-1}[(1+\epsilon_*)/(1+\epsilon_*)(1+\epsilon_i)(1+\epsilon_+)]^{\frac{1}{2}} \leq \exp(4n-3)\epsilon/2.$$

Since the passage from $\vec{B}$ to $\check{B}$ is exact the singular values diminish by τ^2. •

Remark: It can be shown by similar means that one **dqd** transformation cannot alter any singular value by more than $3(n-1)$ *ulps*.

Theorem 4 is much stronger than the familiar error analysis based on norms because:

1. The perturbed matrices considered here inherit the bidiagonal structure
2. The bounds are very much smaller than those from DK (see Section 10) or the Golub-Reinsch algorithm (see Chapter 8 of [11]).

For multiple sweeps of **dqds**, our results can be stated more simply in terms of the positive sequence $\{\check{Z}_l\}$ where l denotes the sweep with $\check{Z}_1 = Z_1 = Z$. See Figure 2 for the corresponding commutative diagram. Then by combining $\check{Z}_l \rightarrow Z_{l+1} \rightarrow \check{Z}_{l+1}$ one obtains that $\check{Z}_{l+1}$ is the exact **dqds** transformation of a perturbed (in relative sense) $\check{Z}_l$. Thus backward stability is present for $\{\check{Z}_l\}$.

Similarly it can be shown that the sequence $\{\vec{Z}_l\}$ is forward stable (in relative sense) with $\vec{Z}_f = Z_f$ where Z_f is the final computed result. On exact application of **dqds** on $\vec{Z}_l$ we get $\check{Z}_{l+1}$ instead of $\vec{Z}_{l+1}$ (see Figure 2) and the error between $\vec{Z}_{l+1}$ and $\check{Z}_{l+1}$ is small.

Example 3 The following experiment shows vividly the difference between an algorithm that obtains high relative accuracy (**dqds**) and one that does not (LINPACK **dsvdc**

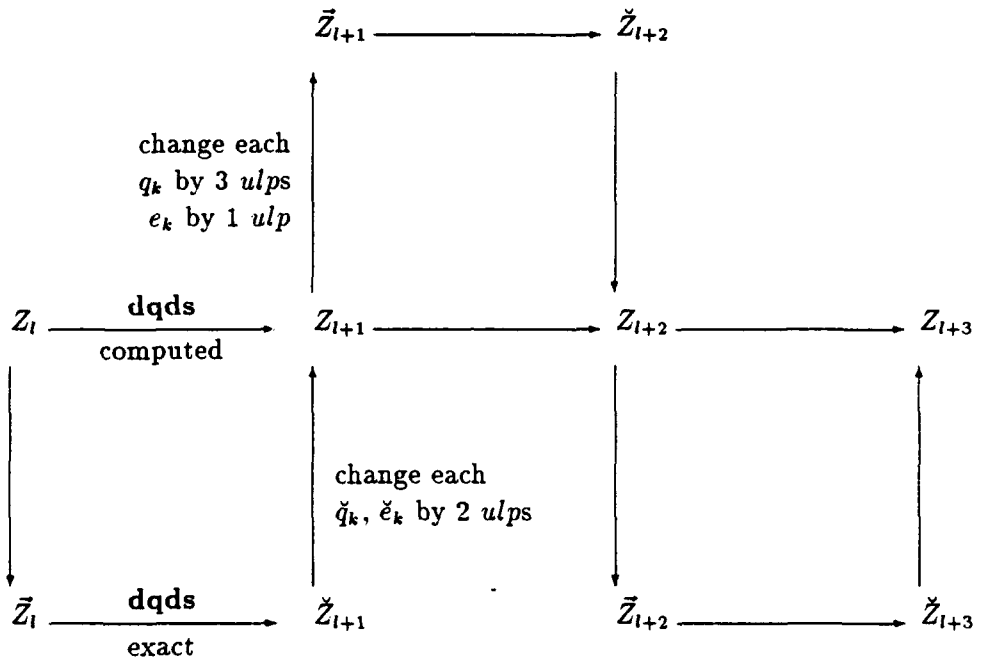


Figure 2: Effects of roundoff for multiple sweeps

based on the Golub-Reinsch algorithm) but which delivers excellent absolute accuracy. We took the graded matrix B_+,

$$a_{i-1} = \beta a_i \quad , \quad b_i = a_i$$

with $a_n = 1$, $n = 8$ and $\beta = 60$. We applied both algorithms to B_+ and its reversal B_-,

$$a_i \rightarrow a_{n+1-i} \quad , \quad b_i \rightarrow b_{n-i}.$$

We did not allow any flipping of the matrix within the **dqds** algorithm although such flipping improves convergence. See next section.

In Tables 5 and 6, the third column shows, abs_i,

$$abs_i := (\sigma_i[B_-] - \sigma_i[B_+])/\sigma_1[B_+]$$

the differences between outputs scaled by the 2-norm of the nicer matrix. Recall that $macheps \approx 2^{-53} \approx 1.1 \times 10^{-16}$. For **dsvdc** (see Table 5), it can be seen that the absolute error is even smaller than absolute stability guarantees.

In Tables 5 and 6, the fourth column shows, rel_i,

$$rel_i := (\sigma_i[B_-] - \sigma_i[B_+])/\sigma_i[B_+]$$

the relative differences in the outputs. For **dqds** the largest magnitude is less than two *macheps* (see Table 5) while for **dsvdc** (see Table 6), it is very much larger and shows that **dsvdc** does not give relative accuracy.

i	$\sigma_i[B_+]$	$\sigma_i[B_-]$	abs_i	rel_i
1	3.9590303657774160D+12	3.9590303657774155D+12	-1.2D-16	-1.2D-16
2	5.7143240472800255D+10	5.7143240472800247D+10	-1.9D-18	-1.3D-16
3	8.9790986853271568D+08	8.9790986853271568D+08	0.0D+00	0.0D+00
4	1.4489876544914651D+07	1.4489876544914651D+07	0.0D+00	0.0D+00
5	2.3661793507020348D+05	2.3661793507020348D+05	0.0D+00	0.0D+00
6	3.8884661685208386D+03	3.8884661685208386D+03	-1.1D-25	-1.2D-16
7	6.4142972113704085D+01	6.4142972113704109D+01	3.6D-27	2.2D-16
8	3.5351579203702068D-01	3.5351579203702068D-01	0.0D+00	0.0D+00

Table 5: Numerical results using **dqds** for Example 3

i	$\sigma_i[B_+]$	$\sigma_i[B_-]$	abs_i	rel_i
1	3.9590303657774146D+12	3.9590303657774160D+12	3.7D-16	3.7D-16
2	5.7143240472800224D+10	5.7143240472800278D+10	1.3D-17	9.3D-16
3	8.9790986853271544D+08	8.9790986853271413D+08	-3.3D-19	-1.5D-15
4	1.4489876544914653D+07	1.4489[illegible]544914989D+07	8.5D-20	2.3D-14
5	2.3661793507020345D+05	2.3661793507022545D+05	5.6D-21	9.3D-14
6	3.8884661685208386D+03	3.8884661685173243D+03	-8.9D-22	-9.0D-13
7	6.4142972113704073D+01	6.4142972113772929D+01	1.7D-23	1.1D-12
8	3.5351579203702068D-01	3.5351579205582154D-01	4.7D-24	5.3D-11

Table 6: Numerical results using **dsvdc** for Example 3

8 Convergence

8.1 Linear Convergence

Convergence of the Cholesky algorithm and of the standard qd algorithm for tridiagonal matrices (in the positive case) have been given by Rutishauser and others (see [22], [27]). Nevertheless we present here a direct convergence proof for **oqd** because it is a new algorithm and the proof is both short and illuminating. We give a brief discussion of the effects of finite precision on the results of the theorem at the end of the section.

The point is that a computed cosine c_k can equal unity even though the corresponding sine s_k may merely be small.

Our proof makes use of the fact that a symmetric tridiagonal matrix with nonzero superdiagonal entries (e.g. B^tB) cannot have multiple eigenvalues [18], [27]. In other words, the singular values of a positive bidiagonal B are distinct

$$\sigma_1 > \sigma_2 > \ldots > \sigma_n. \tag{28}$$

One must bear in mind that distinct σ_i may be equal to working precision.

Theorem 5 (Convergence of oqd) *From a positive bidiagonal B_1 the unshifted* ***oqd*** *algorithm produces a sequences $\{B_l\}_1^\infty$ of orthogonally equivalent bidiagonals. As $l \to \infty$,*

$$B_l \to \Sigma = diag(\sigma_1, \ldots, \sigma_n)$$

Furthermore, if $a_n \leq a_1$, then the sequence $\{\prod_{i=1}^{n-1} b_i^{(l)}\}$ is monotone decreasing in l from the beginning. Each $\{b_i^{(l)}\}_{l=1}^\infty$ converges linearly to 0 with convergence factor σ_{i+1}/σ_i.

Proof: Consider a typical step of **oqd** (Algorithm 1). Since there are no subtractions each B_l is positive since B_1 is positive.

Equation (7) can be written in the form,

$$\hat{a}_k = c_{k-1} a_k / c_k \text{ for } k = 1, \ldots, n \tag{29}$$

provided that we set $c_0 = c_n = 1$. Take the product of the first k instances of (29) to find

$$\prod_{i=1}^{k} \hat{a}_i = (\prod_{i=1}^{k} a_i)/c_k \geq \prod_{i=1}^{n} a_i. \tag{30}$$

The sequence $\{\prod_{i=1}^{k} a_i^{(l)}\}$ is bounded above, by $\|B\|^k$, and monotone increasing by (30) and thus convergent. The limit may be written $\prod_{i=1}^{k} \mu_i$ to reveal that, as $l \to \infty$,

$$a_i^{(l)} \to \mu_i \tag{31}$$

$$c_i^{(l)} \to 1 \tag{32}$$

$$s_i^{(l)} \to 0. \tag{33}$$

By (8) and (33), as $l \to \infty$,

$$b_i^{(l+1)} = s_i^{(l)} a_{i+1}^{(l)} \to 0 \ , \ i = 1, \ldots, n-1 \tag{34}$$

Thus B_l converges to diagonal form and each μ_i is a singular value. To identify μ_i use the product rhombus rule to find

$$b_k^{(l)} / b_k^{(l-1)} = a_{k+1}^{(l-1)} / a_k^{(l)} \to \mu_{k+1}/\mu_k. \tag{35}$$

Since $\{b_k^{(l)}\}$ is bounded (by $\|B_1\|$) the limit in (35) cannot exceed 1. By (28) the limit is not 1 and thus $\mu_{k+1} < \mu_k$, $k = 1, \ldots, n-1$, and we may identify μ_k as σ_k. Thus, (35) proves that $b_k^{(l)} \to 0$ linearly with convergence factor σ_{k+1}/σ_k, as claimed, and $B_l \to \Sigma$.

Finally consider $\prod_{i=1}^{n-1} b_i^{(l)}$. Apply (8) and (6) in turn to find

$$\begin{aligned}
\hat{b}_i \ldots \hat{b}_{n-1} &= b_1 \ldots b_{n-1} \frac{a_2 \ldots a_n}{\hat{a}_1 \ldots \hat{a}_{n-1}} \\
&= b_1 \ldots b_{n-1} \frac{c_1}{a_1} \frac{c_2}{c_1} \frac{c_3}{c_2} \ldots \frac{c_{n-1}}{c_{n-2}} a_n \\
&= b_1 \ldots b_{n-1} \left(\frac{a_n}{a_1} \right) c_{n-1} \\
&< b_1 \ldots b_{n-1},
\end{aligned}$$

if $a_n \leq a_1$. Since B may be flipped about its antidiagonal without altering the singular values there is no loss of generality in assuming that $a_n \leq a_1$. In this case $\prod_{i=1}^{n-1} b_i^{(l)}$ is monotone decreasing with l from the start. •

It is worth mentioning that Rutishauser's proof of convergence for **qd** (Algorithm 3). is based on the observation that $\|\hat{B}_{(k)}\|_F > \|B_k\|_F$ and $\|B_{-k}\|_F < \|B_{-k}\|_F$, $k = 1, 2, \ldots, n$. Here $M_{(\pm k)}$ is the submatrix of M containing the first (last) k columns.

By taking the product of the final $n-k$ instances of (29) one finds that $\{\prod_{i=k}^{n} a_i^{(l)}\}$ is monotone decreasing in l for $k = n-1, n-2, \ldots, 2, 1$. In particular $a_1^{(l)}$ increases and $a_n^{(l)}$ decreases so that a flipping of B is needed at most once.

The assertions of Theorem 5 bear up quite well in finite precision arithmetic. The sequence $\{\prod_{i=1}^{k} a_i^{(l)}\}$ is monotone nondecreasing and so $c_i^{(l)} \to 1$ as $l \to \infty$. Thus in considering a sequence of computed trigonometric values we do not wish to infer $s_k^{(l)} \to 0$ from $c_k^{(l)} \to 1$. So the first casualty is the conclusion that $s_i^{(l)} \to 0$. Instead we find that $b_i^{(l)}$ becomes negligible relative to $a_{i+1}^{(l)}$ and $a_i^{(l)}$. Even so, in the absence of underflow. the diagonal entries eventually rearrange themselves in (almost) monotone nonincreasing order. Though distinct, by (28), some singular values may be equal to working accuracy and diagonal monotonicity may actually fail by one or two *ulp*s (units in the last place held) because the ratio, though exceeding 1 might be too small to cause the neighbouring b-value to grow at all. All in all the practical **oqd** performs as closely to exact **oqd** as it is reasonable to expect.

8.2 Quadratic Convergence

Consider the last few steps in **dqds** with shift τ:

$$\hat{q}_{n-1} = d_{n-1} + e_{n-1}$$

$$\hat{e}_{n-1} = e_{n-1} q_n / \hat{q}_{n-1}$$

$$d_n = d_{n-1} q_n / \hat{q}_{n-1} - \tau^2$$

$$\hat{q}_n = d_n .$$

Hence

$$\begin{aligned} \hat{e}_{n-1}\hat{q}_n &= \frac{e_{n-1}q_n}{\hat{q}_{n-1}} \left[q_n (1 - \frac{e_{n-1}}{\hat{q}_{n-1}}) - \tau^2 \right], \\ &= \frac{e_{n-1}q_n}{\hat{q}_{n-1}} \left[q_n - \tau^2 - \frac{q_n e_{n-1}}{\hat{q}_{n-1}} \right] \end{aligned} \tag{36}$$

In exact arithmetic, as $\tau \to \sigma_n[B_n]$ we have $q_n \to 0$, $e_{n-1} \to 0$, $q_{n-1} \to \sigma_{n-1}^2[B_n] - \sigma_n^2[B_n] := gap > 0$ because the singular values are distinct if the initial B is of positive type. Thus convergence will be quadratic with respect to this *gap*.

Expression (36) shows that if

$$0 \leq q_n - \tau^2 \leq 2\frac{q_n e_{n-1}}{\hat{q}_{n-1}} \tag{37}$$

then

$$|q_n - \tau^2 - \frac{q_n e_{n-1}}{\hat{q}_{n-1}}| \leq \frac{q_n e_{n-1}}{\hat{q}_{n-1}}$$

and so, by (36),

$$\frac{|\hat{e}_{n-1}\hat{q}_n|}{(e_{n-1}q_n)^2} \leq \frac{1}{\hat{q}_{n-1}^2} \rightarrow \frac{1}{gap^2} > 0,$$

as $\tau \rightarrow \sigma_n[B_n]$. Thus (37) shows a (theoretical) interval for those τ that deliver quadratic convergence. Next we seek a usable expression that will ultimately lie in that interval.

The perfect shift is

$$\tau^2 = q_n(1 - \frac{e_{n-1}}{\hat{q}_{n-1}})$$

so that the natural strategy is to estimate $\hat{q}_{n-1}$ from

$$\begin{aligned} \hat{q}_{n-1} &= (1 - \frac{e_{n-2}}{\hat{q}_{n-2}})q_{n-1} - \tau^2 - e_{n-1} \\ &\approx (1 - \frac{e_{n-2}}{q_{n-2}})q_{n-1}, \end{aligned}$$

when e_{n-2} and e_{n-1} are small enough. We may assume that $n > 2$ and so may use

$$\tau^2 = q_n * (1 - e_{n-1}/q_{n-1} * \max\{\frac{1}{2}, 1 - e_{n-2}/q_{n-2}\}) \tag{38}$$

provided that $0 < d_{n-1} = \min_{1 \leq i \leq n-1} d_i$ and $d_n < d_{n-1}$.

8.3 Cubic Convergence

The assertion in Section 8.2 that the shift $\tau^2 = q_n$ yields quadratic convergence for **qds** and **dqds** appears to contradict the result of Rutishauser [23] that this choice yields cubic convergence. See also Rutishauser and Schwarz [26] and Chapter 8 of Wilkinson [29]. Actually, there is no anomaly because the shift strategies are not quite the same. In our terminology what Rutishauser suggests is that the qd transform with $\tau^2 = q_n$ should not be formed explicitly. The only item wanted from it is $\hat{q}_n$ and it is assumed to be the only negative q_i. Then it is shown that $q_n + \hat{q}_n$ is a fourth order approximation to σ_n^2 *from below*. A qd transform of $\{q, e\}$ with shift $\tau^2 = q_n + \hat{q}_n$ will yield cubic convergence.

The point that is stressed by neither Rutishauser nor Wilkinson is that the computation of q_n costs $\mathcal{O}(n)$ operations, very close to 1 step of **qds**. From another perspective Rutishauser's analysis is a disguised derivation of the cubic convergence of the tridiagonal QR algorithm with the Rayleigh quotient shift.

For our algorithm **dqds** Rutishauser's (late failure) shift strategy described above is more appealing. Only the first phrase of our algorithm is needed to compute $\hat{q}_n$ and the cost is about $\frac{2}{3}$ of a **dqds** step. Moreover the positive form is preserved a little longer.

For the sake of completeness we indicate why $q_n + \hat{q}_n$ is a fourth order lower bound when q_n is second order in $q_n e_{n-1}$. The relevant tridiagonal matrix is BB^t and its leading principal $(n-1) \times (n-1)$ submatrix is called V. Recall that u_j is column j of I.

Fact 1: Provided $(V - \sigma_n^2 I)$ is invertible q_n may be written as

$$q_n = \sigma_n^2 + (q_n e_{n-1})^2 u_{n-1}^t (V - \sigma_n^2 I)^{-1} u_{n-1}$$

Fact 2: With shift $\tau^2 = q_n$,

$$\hat{q}_n = -(q_n e_{n-1})^2 u_{n-1}^t (V - q_n I)^{-1} u_{n-1}$$

Conclusion,

$$q_n + \hat{q}_n = \sigma_n^2 + (q_n e_{n-1})^2 u_{n-1}^t \left[(V - \sigma_n^2 I)^{-1} - (V - q_n I)^{-1} \right] u_{n-1}.$$

By Hilbert's second resolvent law,

$$q_n + \hat{q}_n = \sigma_n^2 + (q_n e_{n-1})^2 (\sigma_n^2 - q_n) u_{n-1}^t (V - \sigma_n^2 I)^{-1} (V - q_n I)^{-1} u_{n-1}.$$

Using Fact 1 again,

$$q_n + \hat{q}_n = \sigma_n^2 \widehat{+} (q_n e_{n-1})^4 \left\{ u_{n-1}^t (V - \sigma_n^2 I)^{-1} u_{n-1} u_{n-1}^t (V - q_n I)^{-1} (V - \sigma_n^2)^{-1} u_{n-1} \right\}.$$

The gap conditions ensure that the quantity in { } is $\mathcal{O}(1/gap)$.

In contrast to Rutishauser and Wilkinson, we have made no approximations in our derivation. The result is valid so long as the inverse matrices exist.

9 A Preliminary Implementation

9.1 Choice of Shifts

The standard singular value codes in LINPACK and LAPACK need about 2 QR steps per singular value, in most cases, and that provides a hard target to beat. Moreover one of our qd transformations needs only $\mathcal{O}(n)$ flops and no square roots so we are reluctant to spend $\mathcal{O}(n)$ flops on shift selection.

A strategy used to generate the numerical tests in this paper may not be the best but it is based on the following somewhat surprising observation. The upper bound

$$d_k = \min_i d_i$$

is an increasingly good estimate for $\sigma^2_{min}[\hat{B}]$. Moreover, at each step, we will know the index k from the previous step. This index points to the largest diagonal entry of $(B^tB)^{-1}$ and helps tell us whether σ^2_{min} has yet migrated to the bottom of B^tB. If $k \geq n-1$ we expect the trailing 2×2 principal submatrix of BB^t to give a good approximation to σ^2_{min}. When $k < n-1$ the matrix is not yet in asymptotic form and the situation is more difficult. However we do know that $\sqrt{q_k + e_k}$ is the smallest (leftmost) centre of all the Johnson discs for the matrix of equation (15). If this disc were separated from the rest of the discs then it would certainly contain σ_{min}. Even if it were not isolated this disc might still contain σ_{min} so we use it.

Strictly speaking we need

$$\begin{aligned} (a_k - \frac{1}{2}(b_k + b_{k-1}))^2 &\geq (a_k - \sqrt{e})^2 \\ &> q_k - 2\sqrt{q_k e} + \epsilon \end{aligned} \tag{39}$$

where $e = \max\{e_k, e_{k-1}\}$. When this disc is isolated then the lower bound is definitely too small. A more cautious formula is

$$\min_k \{q_k - \sqrt{q_k e} + e, 0.96 * sup\}$$

Since our estimates may exceed σ_{min} we must face the possibility of rejection of a shift τ^2. How should the new shift be chosen ? On one hand it is not efficient to panic and simply bisect, $\tau^2 \leftarrow \frac{1}{2}(inf + \tau^2)$; on the other hand we must avoid being trapped close to a large overestimate of σ_{min}. Our compromise is to multiply the previous gap between sup and τ^2 by 4 to yield

$$\tau^2 \leftarrow \tau^2 - \min\{4 * (sup - \tau^2), \frac{1}{2}\tau^2\}.$$

9.2 Splitting and Deflation

In Section 2 it was noted that if $e_i = 0$ (i.e. $b_i = 0$), for $i < n$, then the bidiagonal B splits into two complementary subdiagonals. Consider now the case when e_i (or b_i) is small enough to permit such a splitting without making a relative change in any singular value exceeding a given tolerance η. Our situation is a little more complicated than the one studied in DK because of the non-restoring shift. Let σ^2 denote the cumulative sum of all shifts used on the given matrix in the qd algorithm (which computes the squared singular values).

Our criteria are based on Weyl's monotonicity theorem for eigenvalues of symmetric matrices. Consider a bidiagonal B and the possibility of neglecting b_i for a given $i < n$. Write

$$B = \begin{pmatrix} B_1 & b_i \Delta_{i-1} \\ & B_2 \end{pmatrix}$$

where $\Delta_{i,j}$ is a null matrix except that the (i,j) is unity. There are two cases. Case I:

$$B^tB + \sigma^2 I = \begin{pmatrix} B_1^tB_1 + \sigma^2 I & a_ib_i\Delta_{i,1} \\ a_ib_i\Delta_{1,i} & B_2^tB_2 + b_i^2\Delta_{1,1} + \sigma^2 I \end{pmatrix}$$

Case II:

$$BB^t + \sigma^2 I = \begin{pmatrix} B_1B_1^t + b_i^2\Delta_{i,i} + \sigma^2 I & a_{i+1}b_i\Delta_{i,1} \\ a_{i+1}b_i\Delta_{1,i} & B_2B_2^t + \sigma^2 I \end{pmatrix}$$

The spectral norm of the matrix

$$\begin{pmatrix} 0 & \alpha \\ \alpha & 0 \end{pmatrix}$$

is $|\alpha|$ and it is submatrices of this form that we will remove. Weyl's theorem states that on subtracting a principal submatrix of the above form, no eigenvalue is changed by more than $|\alpha|$. Apply this result to Case I and conclude that if

$$a_ib_i < 2\eta(\sigma^2 + \sigma_{min}^2[B_1]) \tag{40}$$

then there are i indices j such that

$$\lambda_j[B^tB + \sigma^2 I] = \lambda_j[B_1^tB_1 + \sigma^2 I](1+\epsilon_j)^2$$

with $\epsilon_j < \eta$. Here $\lambda[M]$ is an eigenvalue of M. Recall that B is not the original matrix whose singular values are to be found. However the eigenvalues of $B^tB + \sigma^2 I$ are the squares of the wanted numbers.

Applying Weyl's theorem to Case II shows that if

$$a_{i+1}b_i < 2\eta(\sigma^2 + \sigma_{min}^2[B_2]) \tag{41}$$

then there are $n - i + 1$ indices j such that

$$\lambda_j[B^tB + \sigma^2 I] = \lambda_j[B_2B_2^t + \sigma^2 I](1+\epsilon_j)^2$$

with $\epsilon_j < \eta$.

Since $\sigma_{min}[B_1]$ and $\sigma_{min}[B_2]$ are not usually known (unless $i = 1, 2,$ or $n-1, n$) the criteria given above must be replaced by a more exacting one

$$b_i < \eta(\sigma^2 + inf)/\max\{a_i, a_{i+1}\} \tag{42}$$

where inf is the best lower bound on $\sigma_{min}^2[B]$.

The conditions (41) and (42) are more severe than DK's

$$b_i < \eta\sigma_{min}[B] \tag{43}$$

when $\sigma^2 < inf$ since $\max a_i, a_{i+1} > \sqrt{inf}$. However after the tiny eigenvalues are found it is not fair to compare b_i with the current B. In principal (42) and (43) can be merged into

$$e_i < \eta^2 \max\left\{inf, 4(\sigma^2 + inf)\left(\frac{\sigma^2 + inf}{\max\{q_i, q_{i+1}\}}\right)\right\} \tag{44}$$

When $i = n-1$ or $n-2$ criterion (40) tells us when to deflate the bottom 1×1 or 2×2 submatrix from B and be sure that the singular values of B_1 are adequate approximations to the remaining singular values of B.

9.3 Performance of a Prototype Implementation

We have developed and implemented **dqds** in FORTRAN 77 to exploit and study the theory we have developed in this paper. This prototype program is built in modular fashion.

We have run our code on a broad test bed of bidiagonals. Here we report on comparisons on three interesting classes using our **dqds** and LINPACK's **dsvdc** (with reduction to bidiagonals removed).

Example 4 (nice matrices) We considered the graded matrix B_+ defined earlier in Example 3 with the parameter $\beta = 2$ and $n = 30$. Table 7 gives the performance of this example and other examples in this section. The speedup is the time taken by **dsvdc** divided by **dqds**.

We have also tested this problem with $n = 40$ and in that case the LINPACK **dsvdc** returned with an error flag as it could not compute σ_{40} within 30 iterations. We were prevented from comparing with larger values of n because **dsvdc** reported errors. •

Example 5 (perversely graded) To make conditions artificially difficult for **dqds**, we also ran the programs with the perversely graded matrix B_- as the input with $n = 30$ and $\beta = 2$. See Table 7 for details.

Usually, our **dqds** will flip B_- to obtain B_+. If the user does not flip B_- before calling **dsvdc** then the time ratio goes up to 18.9.

dsvdc also failed to converge for many combinations of β and n. •

Example 6 Let B_w be the Wilkinson-type bidiagonal matrix where

$$a_i = |i - \frac{n}{2} + 1| \quad , \quad i = 1, \ldots, n$$

$$b_i = 1 \quad , \quad i = 1, \ldots, n-1$$

This matrix has close eigenvalues (twins) and our current coding does not fully exploit this structure. Hence the low performance compared with the previous results. •

Example 7 Doubled Wilkinson-type matrices, B_{2w},

$$a_i = |i - \frac{n}{4} + 1| \quad , \quad i = 1, \ldots, \frac{n}{2}$$

$$a_{i+\frac{n}{2}} = a_i \quad , \quad i = \frac{n}{2} + 1, \ldots, n$$

$$b_i = 1 \quad , \quad i = 1, \ldots, n-1$$

with $n = 41$. This matrix has close singular values (quads) and some of them are exactly equal. Table 7 gives the details. •

Example 8 Toeplitz matrix B_t,

$$a_i = 1 \quad , \quad b_i = 2.$$

For $n = 100$, the matrix has a tiny singular value; others are between 1 and 3. •

Example	Matrix	n	**dqd** sweeps	**dsvdc** sweeps	speedup
4	B_+	30	52	60	10.2
5	B_-	30	79	101	12.3
6	B_w	21	78	62	4.8
7	B_{2w}	41	230	120	4.8
8	B_t	100	374	308	11.0

Table 7: Performance comparison

10 The Demmel/Kahan Paper

We summarize the highlights of this impressive contribution [4].

10.1 High Relative Accuracy

Corollary 2 of Theorem 2 of DK. Suppose that $(B+\delta B)_{i,i} = \alpha_{2i-1}a_i$, $(B+\delta B)_{i,i+1} = \alpha_{2i}b_i$. $\alpha_i \neq 0$. Define

$$\bar{\alpha} := \prod_{i=1}^{2n-1} \max\{|\alpha_i|, |\alpha_i^{-1}|\}.$$

Let $\sigma_1' \geq \sigma_2' \geq \ldots \geq \sigma_n'$ be the singular values of $B + \delta B$. Then

$$\sigma_i/\bar{\alpha} \leq \sigma_i' \leq \sigma_i\bar{\alpha} \quad , \quad i = 1, 2, \ldots, n.$$

This shows that bidiagonal matrices determine their singular values to high relative accuracy.

10.2 Bounds for σ_n

It is possible to compute $\|B^{-1}\|_\infty$ and $\|B^{-1}|_1$ using $2(n-1)$ divisions and multiplications. The algorithm is

$$\lambda_j := a_j(\lambda_{j+1}/(\lambda_{j+1} + b_j)) \quad , \quad j = n-1, n-2, \ldots, 1 \quad \text{with} \quad \lambda_n := a_n$$

$$\mu_{j+1} := a_{j+1}(\mu_j/(\mu_j + b_j)) \quad , \quad j = 1, 2, \ldots, n-1 \quad \text{with} \quad \mu_1 := a_1$$

$$\|B^{-1}\|_\infty = 1/\min_j \lambda_j$$

$$\|B^{-1}\|_1 = 1/\min_j \mu_j.$$

Finally,

$$n^{-1/2}\max\{\|B^{-1}\|_\infty^{-1}, \|B^{-1}\|_1^{-1}\} \le \sigma_n \le n^{1/2}\min\{\|B^{-1}\|_\infty^{-1}, \|B^{-1}\|_1^{-1}\}$$

and

$$\min\{\|B^{-1}\|_\infty^{-1}, \|B^{-1}\|_1^{-1}\} \le \sigma_n$$

10.3 A Stopping Criterion

Let $\eta << 1$ be the desired relative accuracy of the computed singular values. Then if either

$$b_j/\lambda_{j+1} \le \eta \quad \text{or} \quad b_j/\mu_j \le \eta$$

set b_j to zero and the two pieces into which B splits may be processed separately. The criteria used in LINPACK [5] can sometimes deliver a zero singular value when it should not and can sometimes fail to suppress a negligible off diagonal entry b_j.

10.4 Bidiagonal QR with Zero Shift

The standard Golub/Reinsch algorithm [11], [10] used in LINPACK may be simplified when no shifts are used. Of more importance is the fact that in this case all round off errors arise multiplicatively. Moreover for the calculation of tiny singular values zero is a good shift and it pays to compute them first rather then letting the standard shift strategy dictate the order in which the singular values are found. The arithmetic effort in the innermost loop is

Golub/Reinsch: 2 calls to ROT + 12 multiplications + 4 additions

Demmel/Kahan: 2 calls to ROT + 4 multiplications.

The procedure ROT computes the sine and cosine needed for a plane rotation using 2 divisions, 3 multiplications, and 1 square root. Here is the algorithm.

$$\begin{aligned}
&oldcs := cs := 1\\
&\text{for } i := 1, n-1\\
&\quad \text{call } ROT(a_i * cs, b_i, cs, sn, r)\\
&\quad \text{if } (i \neq 1)\ b_{i-1} := oldsn * r\\
&\quad \text{call } ROT(oldcs * r, b_{i+1} * sn, oldcs, oldsn, b_i)\\
&\quad h := a_n * cs;\ \ b_{n-1} := h * oldsn;\ \ a_n := h * oldcs\\
&\text{end for}
\end{aligned} \tag{45}$$

In the absence of underflow the error bound on singular values after one zero shift bidiagonal QR transform is

$$|\sigma_i - \sigma'| \leq \frac{w}{1-w}\sigma_i \quad i = 1, \ldots, n$$

where

$$w := 69n^2\epsilon < 1.$$

See Theorem 6, p. 906.

10.5 The Overall Algorithm

if (roundoff in any shift exceeds $tol * bound$ on σ_n) then
  use zero-shift QR or QL
else
  use shifted QR or QL
end if.

10.6 Other Improvements

The new code uses either QL or QR as appropriate according to the way B is graded.

An efficient accurate subroutine is provided to return the singular values and vectors of 2×2 bidiagonal matrices.

Deflation when a diagonal entry a_i vanishes is automatic and occurs at the bottom or top of B.

11 Evolution of qd

Some of the available presentations of the qd-algorithm, see [22], [27], [12] show its close connection with factorization of tridiagonal matrices but some do not [13], [28]. Nevertheless its discovery had nothing to do with matrix decompositions and a knowledge of the origins helps us to understand the somewhat neglected status of the algorithm. In the next few paragraphs we sketch an earlier paper [17] which described the gradual evolution of the qd-algorithm.

The story begins with Daniel Bernoulli in 1728 when he showed that the largest and the smallest roots of an nth degree polynomial can be obtained by iterating an nth degree difference equation. See [2]. The work of Bernoulli was extended by Euler in 1748. See Chapter 17 of [7] (English translation [6]).

We are given a rational function of a complex variable z,

$$f(z) = \sum_{k=0}^{\infty} h_k z^k \quad ,$$

assumed to be regular (analytic) at both $z = 0$ and $z = \infty$. The Taylor series converges to $f(z)$ only within a circle (in $\mathcal{C}$) centred at $z = 0$ and extending up to the nearest pole p_1. However, by analytic continuation, the Taylor coefficients $\{h_k\}$ actually define a unique rational function f on all of $\mathcal{C}$ except the poles $p_1, p_2, p_3, \ldots$. The problem is to determine the poles directly from the $\{h_k\}$ without having recourse to analytic continuation.

In 1884 König [16] showed that if p_1 is a simple pole and smaller than all the others then

$$\lim_{k \to \infty} (h_k / h_{k+1}) = p_1.$$

Exactly one hundred years ago (i.e. in 1892), in his dissertation, J. Hadamard showed that

$$\lim_{k \to \infty} \left(\frac{H_m^{k+1}}{H_m^k} \right) = \prod_{i=1}^{m} p_i$$

where

$$H_m^k = \det \begin{bmatrix} h_k & h_{k+1} & \cdots & h_{k+m-1} \\ h_{k+1} & h_{k+2} & \cdots & h_{k+m} \\ & & \cdots & \\ h_{k+m-1} & h_{k+m} & \cdots & h_{k+2m-1} \end{bmatrix}$$

The H_m^k are called Hankel determinants. It follows that

$$p_m = \lim_{k \to \infty} \left(\frac{H_m^{k+1} / H_{m-1}^{k+1}}{H_m^k / H_{m-1}^k} \right).$$

The solution is brilliant but does not give us a practical algorithm.

During the 1920s, in Scotland, A. C. Aitken rediscovered for himself a remarkable connection among Hankel matrices that was known to Hadamard but not exploited by him;

$$(H_m^k)^2 + H_{m+1}^{k-1} H_{m-1}^{k+1} = H_{m+1}^{k-1} H_{m-1}^{k+1}. \tag{46}$$

The relation (46) permits the computation of all the H_m^k without being drowned in determinantal equations.

The blemish in (46) is that the H_m^k are not of direct interest. We want to compute

$$q_m^{(k)} := \frac{H_m^{k+1} / H_m^k}{H_{m-1}^k / H_{m-1}^k}.$$

$$
\begin{array}{ccccc}
q_1^{(0)} & & & & \\
& e_1^{(0)} & & & \\
q_1^{(1)} & & q_2^{(0)} & & \\
& e_1^{(1)} & & e_2^{(0)} & \\
q_1^{(2)} & & q_2^{(1)} & & \cdot \\
& e_1^{(2)} & & e_2^{(1)} & \\
q_1^{(3)} & & q_2^{(2)} & & \cdot \\
& e_1^{(3)} & & e_2^{(2)} & \\
\cdot & & q_2^{(3)} & & \cdot \\
& \cdot & & e_2^{(3)} & \\
\cdot & & \cdot & & \cdot
\end{array}
$$

Figure 3: qd in a (modified) difference table

Rutishauser's clever observation was that if one introduces

$$e_m^{(k)} := \frac{H_{m-1}^{k+1} H_{m+1}^{k}}{H_m^k H_m^{k+1}}$$

then (46) implies

$$q_m^{(k)} + e_m^{(k)} = q_m^{(k+1)} + e_{m-1}^{(k+1)},$$

the additive rhombus rule, while the definitions of q and e give the product rhombus rule

$$q_m^{(k+1)} q_m^{(k+1)} = q_{m+1}^{(k)} e_m^{(k)}.$$

The rhombus rules were introduced at the end of Section 4. The qs and es are best laid out in a tableau that is like a difference table. See Figure 3.

This qd table may be built up via the rhombus rules either from column 1 or from the top diagonal. The first column, $\{q_1^{(k)}\}$ is at hand, since

$$q_1^{(k)} = H_1^k / H_1^{k-1} = h_k / h_{k-1} \ , \ \ k \geq 1$$

and

$$e_1^{(k)} = q_1^{(k+1)} - q_1^{(k)}.$$

This is far simple than Hadamard's solution but, in finite precision arithmetic, it is hopelessly unstable because the later e's are (modified) differences of converging values.

Fortunately computation along descending diagonals is stable but here the difficulty is the calculation of the top diagonal. This is not as daunting as it appears at first. If the

function $f(\zeta)$ has only n poles then all q (and e) columns beyond the nth vanish. Then it suffices to build the $n \times n$ Hankel matrix H_n^0 and compute its triangular factorization

$$H_n^0 = L_n D_n L_n^t$$

where $D_n = diag(d_1, \ldots, d_n)$ holds the successive pivots. It turns out that

$$q_k^{(0)} = H_k^0 / H_{k-1}^0 = d_k \quad , \quad k = 1, \ldots, n$$

The $e_k^{(0)}$ are found from the pivots of H_n^1. This is the practical way to compute the poles from the Taylor coefficients. In fact a careful form of row interchanges (not partial pivoting) may be used to improve the accuracy of the factorization.

Next we relate the qd tableau to the computation of eigenvalues. Given a square matrix C the appropriate rational function f comes from the resolvent,

$$f(z) = x^t (I - zC)^{-1} y,$$

where x and y are column vectors. A technical assumption is needed to guarantee that the qd tableau is well defined. In the language of control theory, see [15], the linear dynamical system $\mathcal{S}(C, y, x^t)$ must be minimal. If it is not minimal, then we might not be able to find all the poles of the system.

For this function f,

$$h_k = x^t C^k y / x^t C^{k-1} y$$

as so the $\{h_k\}$ could be computed by the power method. However, it would be preferable to compute $\{q_1^{(0)}, e_1^{(0)}, q_2^{(0)}, e_2^{(0)}, \ldots\}$ directly from C and we now know that this can be done by invoking the Lanczos algorithm on C and using the resulting tridiagonal matrix J. It turns out that the pivots that occur in computing the triangular factorization of J are the $\{q_k^{(0)}\}$ and their reciprocals are the $\{e_k^{(0)}\}$. The details are given in [17]. In other words,

$$J = \begin{bmatrix} 1 & & & & & \\ e_1^{(0)} & 1 & & & & \\ & e_2^{(0)} & 1 & & & \\ & & e_3^{(0)} & 1 & & \\ & & & \ddots & \ddots & \\ & & & & \ddots & \end{bmatrix} \begin{bmatrix} q_1^{(0)} & 1 & & & & \\ & q_2^{(0)} & 1 & & & \\ & & q_3^{(0)} & 1 & & \\ & & & q_4^{(0)} & & \\ & & & & \ddots & \ddots \\ & & & & & \ddots \end{bmatrix}.$$

We see here how the LR algorithm on tridiagonals was hidden in the qd table.

12 The Continued Fraction Connection

There is an intimate connection between our bidiagonal matrix B, the tridiagonal matrix $T = B^t B$, and a continued fraction associated with them. Properties of continued fractions influenced the qd algorithm initially and only later did the LR transformation

emerge and nearly displace the continued fraction. We can not find any discussion by Rutishauser of the connection between the continued fraction and B^tB so we supply it here.

Recall the notation from Sections 2, 3, and 4.

$$B = bidiag\left\{ \begin{array}{ccccccccc} & b_1 & & b_2 & . & & b_{n-2} & & b_{n-1} & \\ a_1 & & a_2 & & . & . & & a_{n-1} & & a_n \end{array} \right\}.$$

$$q_i = a_i^2$$

$$e_i^2 = b_i^2 \ , \ e_0 = e_n = 0$$

$$T = \text{tridiag}\left\{ \begin{array}{ccccccccc} & \sqrt{q_1e_1} & & \sqrt{q_2e_2} & & \sqrt{q_3e_3} & & \sqrt{q_{n-1}e_{n-1}} & \\ q_1 & & q_2+e_1 & & q_3+e_2 & & \cdots & & q_n+e_{n-1} \\ & \sqrt{q_1e_1} & & \sqrt{q_2e_2} & & \sqrt{q_3e_3} & & \sqrt{q_{n-1}e_{n-1}} & \end{array} \right\}.$$

Rutishauser associates with T the continued fraction

$$F(\zeta) = \frac{1}{\zeta - q_1 -} \ \frac{e_1q_1}{\zeta - q_2 - e_1 -} \ \frac{e_2q_2}{\zeta - q_2 - e_2 -} \ \cdots \qquad (47)$$

It is not obvious how $F(\zeta)$ relates to T. The answer is

$$F(\zeta) = [(\zeta I - T)^{-1}]_{1,1}$$

or more generally,

$$F(\zeta) = x^t(\zeta I - T)^{-1}y$$

with x and y as defined near the end of Section 11. The inverse is well defined for all ζ with $|\zeta|$ exceeding the spectral radius of T. The particular form of the continued fraction arises from the triangular factorization of $\zeta I - T$ from the bottom up:

$$\zeta I - T = \tilde{L}^t\tilde{D}\tilde{L}$$

where $\tilde{L}$ is unit triangular and

$$\tilde{D} = diag(\tilde{d}_1, \tilde{d}_2, \ldots, \tilde{d}_n)$$

$$\tilde{d}_i = \tilde{d}_i(\zeta).$$

Then

$$(\zeta I - T)^{-1} = \tilde{L}^{-1}\tilde{D}^{-1}\tilde{L}^{-t}$$

and

$$F(\zeta) = \tilde{d}_1^{-1}$$

The recurrence for the $\tilde{d}_j$ is

$$\begin{aligned} \tilde{d}_n &:= \zeta - q_n - e_{n-1} \\ \tilde{d}_j &:= \zeta - q_j - e_{j-1} - q_je_j/\tilde{d}_{j+1} \text{ for } j = n-1, \ldots, 2, 1. \end{aligned} \qquad (48)$$

This establishes (47).

There is a simpler continued fraction expansion for $F(\zeta)$. It corresponds to a recurrence for $\tilde{d}_j + e_{j-1}$. From (48)

$$\begin{aligned} \tilde{d}_j + e_{j-1} &= \zeta - q_j(1 + \frac{e_j}{\tilde{d}_{j+1}}) \\ &= \zeta - q_j / \left(\frac{\tilde{d}_{j+1} + e_j - e_j}{\tilde{d}_{j+1} + e_j} \right) \\ &= \zeta - q_j / \left(1 - e_j/(\tilde{d}_{j+1} + e_j) \right) \end{aligned} \tag{49}$$

Since $e_0 = 0$, (49) gives

$$F(\zeta) = 1/\tilde{d}_1 = \frac{1}{\zeta -} \; \frac{q_1}{1 -} \; \frac{e_1}{\zeta -} \; \frac{q_2}{1 -} \; \frac{e_2}{\zeta -} \; \cdots$$

This form is remarkable for the direct connection of q_i and e_i to the $(1,1)$ entry of $(\zeta I - T)^{-1}$.

References

[1] F. L. Bauer. qd-method with Newton shift. Technical Report 56, Computer Science Department, Stanford University, 1967.

[2] Daniel Bernoulli. Observationes de seriebus quae formantur ex additione vel substractione quacunque terminorum su mutuo consequentium, ubi praesertim earundem insignis usus pro inveniendis radicum omnium aequationum algebraicarum ostenditur. *Commentarii Academiae Scientiarum Imperialis Petropolitanae*, 3:85–100, 1732 (1728).

[3] B. Codenotti and M. Leoncini. *Parallel Complexity of Linear System Solution.* World Scientific, Singapore, 1991.

[4] James Demmel and W. Kahan. Accurate singular values of bidiagonal matrices. *SIAM J. Sci. Sta. Comput.*, 11:873–912, 1990.

[5] J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W. Stewart. *LINPACK User' Guide.* SIAM, Philadelphia, 1979.

[6] Leonard Euler. *Introduction to Analysis of the Infinite*, volume 1. Springer-Verlag, New York, 1988.

[7] Leonhardo Eulero. *Introductio in Analysin Infinitorum*, volume 8 of *series 1.* Marcum-Michaelem Bousquet, Lausannæ, 1748.

[8] K. Vince Fernando and Beresford N. Parlett. Orthogonal Cholesky algorithm. Technical Report under preparation, Department of Mathematics, University of California at Berkeley, 1992.

[9] K. Vince Fernando and Beresford N. Parlett. qd algorithms for advanced architectures. Technical Report under preparation, Department of Mathematics, University of California at Berkeley, 1992.

[10] Gene H. Golub and C. Reinsch. Singular value decomposition and least squares solutions. *Numer. Math.*, 14:403–420, 1970.

[11] Gene H. Golub and Charles F. Van Loan. *Matrix Computations.* Johns Hopkins University Press, Baltimore, 1989.

[12] P. Henrici. The quotient-difference algorithm. *Nat. Bur. Standards Appl. Math. Series*, 19:23–46, 1958.

[13] A. S. Householder. *The Numerical Treatment of a Single Nonlinear Equation.* McGraw-Hill, New York, 1970.

[14] Charles. R. Johnson. Inversion of matrices by biorthogonalization and related results. ***Linear Algebra and Its Applications***, 112:1–7, 1989.

[15] Thomas Kailath. *Linear Systems.* Prentice-Hall, Englewood Cliffs, NJ, 1980.

[16] Julius König. Ueber eine Eigenschaft der Potenzreihen. *Math. Ann.*, 23:447–449, 1884.

[17] Beresford. N. Parlett. The development and use of methods of LR type. *SIAM Review*, 6:275–295, 1964.

[18] Beresford. N. Parlett. *The Symmetric Eigenvalue Problem.* Prentice-Hall, Englewood Cliffs, NJ, 1980.

[19] C. Reinsch and F. L. Bauer. Rational QR transformation with Newton shift for symmetric tridiagonal matrices. *Numer. Math.*, 11:264–272, 1968.

[20] H. Rutishauser. Der quotienten-differenzen-algorithmus. *Z. Angew. Math. Phys.*, 5:233–251, 1954.

[21] H. Rutishauser. Ein infinitesimales analogon zum quotienten-differenzen-algorithmus. *Arch. Math.*, 5:132–137, 1954.

[22] H. Rutishauser. Solution of eigenvalue problems with the LR-transformation. *Nat. Bur. Standards Appl. Math. Series*, 49:47–81, 1958.

[23] H. Rutishauser. Über eine kubisch konvergente Variante der LR-Transformation. *Z. Angew. Math. Mech.*, 11:49–54, 1960.

[24] H. Rutishauser. *Vorlesungen über numerische Mathematik.* Birkhäuser, Basel, 1976.

[25] H. Rutishauser. *Lectures on Numerical Mathematics.* Birkhäuser, Boston, 1990.

[26] H. Rutishauser and H. R. Schwarz. The LR transformation method for symmetric matrices. *Numer. Math.*, 5:273–289, 1963.

[27] H. R. Schwarz, H. Rutishauser, and E. Stiefel. *Numerical Analysis of Symmetric Matrices.* Prentice-Hall, Englewood Cliffs, NJ, 1973.

[28] G. W. Stewart. On a companion operator for analytic functions. *Numer. Math.*, 18:26–43, 1971.

[29] J. H. Wilkinson. *The Algebraic Eigenvalue Problem.* Clarendon Press, Oxford, 1965.

Legal Notice